



# Lynch-Morawska Systems on Strings

**Daniel S. Hono II**  
**Paliath Narendran**  
**Rafael Veras**

# Lynch-Morawska Systems on Strings

Daniel S. Hono II<sup>1</sup> and Paliath Narendran<sup>1</sup> and Rafael Veras<sup>1</sup>

<sup>1</sup> University at Albany–SUNY (USA), e-mail: {dhono, pnarendran, rveras}@albany.edu

## Abstract

We investigate properties of convergent and forward-closed string rewriting systems in the context of the syntactic criteria introduced in [8] by Christopher Lynch and Barbara Morawska (we call these *LM*-Systems). Since a string rewriting system can be viewed as a term-rewriting system over a signature of purely monadic function symbols, we adapt their definition to the string rewriting case. We prove that the subterm-collapse problem for convergent and forward-closed string rewriting systems is effectively solvable. Therefore, there exists a decision procedure that verifies if such a system is an *LM*-System. We use the same construction to prove that the *cap problem* from the field of cryptographic protocol analysis, which is undecidable for general *LM*-systems, is decidable when restricted to the string rewriting case.

## 1 Introduction

In this paper we investigate the properties of convergent and forward-closed string rewriting systems. Our motivation comes from the syntactic criteria defined by Christopher Lynch and Barbara Morawska in [8]. They showed that for any term-rewriting system  $R$  that satisfies their criteria (which we call *LM*-Systems), the unification problem modulo  $R$  is solvable in polynomial time. In [6] it was shown that these conditions are tight, i.e., relaxing any of them leads to NP-hard unification problems. It was also shown in [6] that the subterm-collapse problem for term-rewriting systems that satisfy all of the other conditions of *LM*-Systems is undecidable.

In this current work, we show that the subterm-collapse problem is decidable when restricted to convergent and forward-closed string rewriting systems. These string rewriting systems can be viewed as term rewriting systems over a signature of purely monadic function symbols. We give an analogous definition of *LM*-Systems for string rewriting systems. Thus, given a forward-closed and convergent string rewriting system  $T$  there is an algorithm that decides if  $T$  is an *LM*-System.

The construction used to show the decidability of the subterm-collapse problem for forward-closed and convergent string rewriting systems is also used to show that the *cap problem*, an important problem from the field of cryptographic protocol analysis [1], is also decidable for such string rewriting systems. This is in contrast with some of our recent work that shows that the *cap problem*, which is undecidable in general, remains undecidable when restricted to general *LM*-Systems.

## 2 Definitions

We present here some notation and definitions. Only a few essential definitions are given here; for more details, the reader is referred to [3] for term rewriting systems, and to [4] for string rewriting systems.

Let  $\Sigma$  be a finite alphabet. As is usual,  $\Sigma^*$  stands for the set of all strings over  $\Sigma$ . The empty string is denoted by  $\lambda$ . For a string  $x$ ,  $|x|$  denotes its length and  $x^R$  denotes its reversal. A string  $u$  overlaps with a string  $v$  iff there is a non-empty *proper* suffix of  $u$  which is a prefix of  $v$ . For instance, *aba* overlaps with *acc*, but *aba* does not overlap with *cca*. However, *aba* overlaps with itself since *a* is both a prefix and a suffix of *aba*. (See Fig 1.)

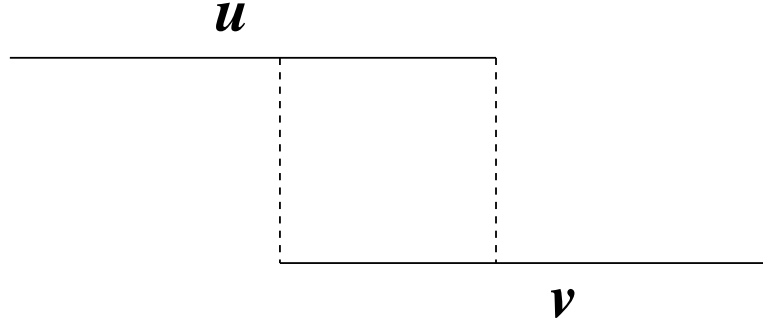


Figure 1: overlap

A string rewriting (rewrite) system (SRS)  $R$  over this alphabet is a set of *rewrite rules* of the form  $l \rightarrow r$  where  $l, r \in \Sigma^*$ ;  $l$  and  $r$  are respectively called the left- and right-hand-side (*lhs* and *rhs*) of the rule. The *rewrite relation* on strings defined by the rewrite system  $R$ , denoted  $\rightarrow_R$ , is

$$\{(xly, xry) \mid x, y \in \Sigma^* \text{ and } (l, r) \in R\}$$

The reflexive and transitive closure of this relation is  $\rightarrow_R^*$ . An SRS  $R$  is *terminating* iff there is no infinite chain of strings  $s_i$ ,  $i \in \mathbb{N}$ , such that  $s_i$   $R$ -rewrites to  $s_{i+1}$ , that is to say  $s_i \rightarrow_R s_{i+1}$ . An SRS  $R$  is *confluent* iff for all strings  $t, s_1, s_2$  such that  $s_1 \xleftarrow{*}_R t \xrightarrow{*}_R s_2$  there exists a string  $t'$  such that  $s_1 \xrightarrow{*}_R t' \xleftarrow{*}_R s_2$ . An SRS  $R$  is *convergent* iff it is both terminating and confluent.

A string is *irreducible* with respect to  $R$  iff no rule of  $R$  can be applied to it. The set of strings that are irreducible modulo  $R$  is denoted by  $IRR(R)$ . Note that this set is a regular language, since  $IRR(R) = \Sigma^* \setminus \{\Sigma^* l_1 \Sigma^* \cup \dots \cup \Sigma^* l_m \Sigma^*\}$ , where  $l_1, \dots, l_m$  are the lhs of the rules in  $R$ . A string  $w'$  is an  $R$ -*normal form* (or a *normal form* if the rewrite system is obvious from the context) of a string  $w$  for an SRS  $R$  if and only if  $w \rightarrow_R^* w'$  and  $w'$  is irreducible. We write this as  $w \rightarrow_R^! w'$ . An SRS  $R$  is *right-reduced* if every right-hand side is in normal form. An SRS  $T$  is said to be *canonical* if and only if it is convergent and *inter-reduced*, i.e., it is right-reduced and, besides, no lhs is a substring of another lhs.

Given a rewrite system  $R$  and a set of strings  $L$ ,  $R^*(L)$  is the set of all descendants of strings from  $L$ , i.e.,  $\{x \mid \exists y \in L : y \rightarrow^* x\}$ , and  $R^!(L)$  the set of normal forms of strings in  $L$  for the rewrite system  $R$ . Thus  $R^!(L) = R^*(L) \cap IRR(R)$ .

String rewriting systems can be viewed as a restricted class of term rewriting systems where all functions are unary. As in [2] a string  $u$  over a given alphabet  $\Sigma$  is viewed as a term over one variable derived from the *reversed* string of  $u$ ; i.e., if  $g, h \in \Sigma$ , the string  $gh$  corresponds to the term  $h(g(x))$ . (In other words, the unary operators defined by the symbols of a string are applied successively in the order in which these symbols appear in that string.) A string of the form  $wl$  where  $w \in \Sigma^*$  and  $l$  is a left-hand side is called a *redex*. A redex is *innermost* if no proper prefix of it is a redex. The longest suffix of an innermost redex that is a left-hand side in  $R$  is called its *l-part* and the remaining prefix is referred to as its *s-part*.

We will also need a special kind of normal form for strings, modulo any given SRS  $T$ . With that purpose, we define, following Sénizergues [9], a *leftmost-largest* reduction as follows: let  $\succ$  be a given total ordering on the alphabet  $\Sigma$  and  $\succ_L$  be its length + lexicographic extension<sup>1</sup>. A rewrite step  $xly \rightarrow xry$  is *leftmost-largest* if and only if (a)  $xl$  is an innermost redex, (b) any other left-hand side that is a suffix of  $xl$  is a suffix of  $l$  as well, (i.e.,  $l$  is the  $l$ -part of this redex) and (c) if  $l \rightarrow r'$  is another rule in the rewrite system, then  $r' \succ_L r$ . A string  $w'$  is said to be a *leftmost-largest (ll-) normal form* of a string  $w$  iff  $w \rightarrow^! w'$  using only leftmost-largest rewrite steps. Given a terminating system  $T$ , it holds that any string  $w$  has a *unique* normal form produced by leftmost-largest rewrite steps alone, since every rewrite step is unique; this unique normal form will be denoted as  $\rho_T(w)$ .

Next, we define what it means for a string  $x \in \Sigma^+$  to cause a subterm collapse.

**Definition 2.1.** Let  $R$  be a convergent string rewriting system. A string  $x$  is said to *cause a subterm-collapse* if and only if there is a non-empty string  $y$  such that  $xy \rightarrow_R^* x$ .

Throughout the rest of the paper,  $a, b, c, \dots, h$  will denote elements of the alphabet  $\Sigma$ , and strings over  $\Sigma$  will be denoted as  $l, r, u, v, w, x, y, z$ , along with subscripts and superscripts.

A string rewrite system  $T$  is said to be *forward-closed* iff every innermost redex can be reduced to its normal form *in one step*.

We now give some preliminary results on convergent and forward-closed string rewriting systems. This first lemma shows that reducing all right-hand sides of rules in  $R$  will preserve the equivalence generated by  $R$  as well as the properties that we are interested in.

**Lemma 2.2.** Let  $R$  be a convergent and forward-closed string rewriting system, and let  $l \rightarrow r$  be a rule in  $R$ . Then  $(R \setminus \{l \rightarrow r\}) \cup \{l \rightarrow \rho_R(r)\}$  is convergent, forward-closed and equivalent to  $R$ .

*Proof.* Let  $R' = (R \setminus \{l \rightarrow r\}) \cup \{l \rightarrow \rho_R(r)\}$  where  $R$  is convergent and forward-closed. We make a few observations first. First of all, since  $R'$  contains the same left-hand sides as  $R$ ,  $IRR(R') = IRR(R)$ . The set of redexes are the same too. Besides, it is not hard to see that  $\rightarrow_R^* \subseteq \rightarrow_{R'}^*$  since  $l \rightarrow_R r \rightarrow_R^* \rho_R(r)$  for all rules  $l \rightarrow r$  in  $R$ .

We first show that  $R'$  and  $R$  are equivalent. This is straightforward since for every rule  $l \rightarrow r \in R$ ,  $l$  and  $r$  are joinable modulo  $R'$  and vice versa. Thus  $\leftrightarrow_R^* = \leftrightarrow_{R'}^*$ .

We next show that  $R'$  is terminating given that  $R$  is convergent. For the sake of deriving a contradiction, assume that  $R'$  is not terminating. Then  $\exists t \in \Sigma^* : (t_i)_{i=0}^\infty$  and  $t_i \rightarrow_{R'} t_{i+1}$  where  $t_0 = t$ . Consider any  $t_i \rightarrow_{R'} t_{i+1}$  step in the above sequence. Then, by definition of reduction, there must be a rule  $l \rightarrow r \in R'$  such that:

$$t_i = xly \rightarrow xry = t_{i+1}$$

Since no left-hand sides of rules in  $R$  were altered in the construction of  $R'$ , we can apply a corresponding rule in  $R$ . If the rule  $l \rightarrow \rho_R(r)$  was used, then we could replace the above step with at most two reduction steps. Thus, we could construct an infinite descending chain modulo  $R$ , which is a contradiction.

Next, we show that  $R'$  is confluent. Suppose it is not. Then since  $R'$  is terminating, there must be a string  $t$  with two distinct normal forms  $t_1$  and  $t_2$ . But since  $R$  is confluent and equivalent to  $R'$ , one of  $t_1$  and  $t_2$  must be *reducible* modulo  $R$ . This is clearly a contradiction since  $IRR(R) = IRR(R')$ .

Thus,  $R'$  is convergent given that  $R$  is convergent.

---

<sup>1</sup>Sénizergues refers to this as the *short-lex* ordering

It remains to show that  $R'$  is forward-closed. For this it is enough to show that every innermost redex can be reduced to its normal form in a *single* reduction step. Let  $x = x'l$  be an innermost redex modulo  $R'$  where  $x, x' \in \Sigma^*$ . Then  $x$  is also an innermost redex modulo  $R$ . Since  $R$  is forward-closed  $x'r \in \text{IRR}(R)$  for  $l \rightarrow r \in R$ . Thus,  $x'r \in \text{IRR}(R')$  as well, again since  $\text{IRR}(R) = \text{IRR}(R')$ .  $\square$

We next show that no left-hand sides of rules of a forward closed and convergent string-rewriting system can be the same.

**Corollary 2.3.** *Let  $R$  be a convergent, forward-closed and right-reduced string rewriting system. Then no two distinct rules have the same left-hand side.*

*Proof.* Suppose not. Let  $l_i \rightarrow r_i \in R$  for  $i \in \{1, 2\}$  such that  $l_1 = l_2$  but  $r_1 \neq r_2$ , but then  $l \rightarrow r_1$  and  $l \rightarrow r_2$  as trivial reductions would not be joinable, as  $r_1$  and  $r_2$  are in normal form.  $\square$

The next preliminary result shows that we can use leftmost-largest reduction steps to reduce an innermost redex to its normal form in a single step.

**Lemma 2.4.** *Let  $R$  be a convergent, forward-closed and right-reduced string rewriting system, and let  $w$  be an innermost redex. Then  $w \rightarrow \rho_R(w)$ , i.e.,  $w$  reduces to its normal form in one leftmost-largest reduction step.*

*Proof.* Let  $w \in \Sigma^*$  be an innermost redex. Then  $w = w'l$  for  $w' \in \Sigma^*$  and by forward closure there must be some rule  $l \rightarrow r \in R$  such that  $l \rightarrow r$  reduces  $w$  to its normal form in a single step. If this were not a leftmost-largest reduction, then there must be some other rule  $l' \rightarrow r' \in R$  such that  $w = w''l'$  is also an innermost-redex. By Corollary 2.3,  $l$  must be a proper suffix of  $l'$  and  $l'$  must be unique, then  $w \rightarrow w''r' \in \text{IRR}(R)$  and  $w \rightarrow w'r \in \text{IRR}(R)$ , which contradicts the convergence of  $R$ .  $\square$

### 3 LM-Conditions for String Rewriting Systems

We now give an equivalent definition of *quasi-determinism* for string rewriting systems  $R$ . This definition is adapted from that of [8]. We also define a *right-hand side critical pair* for string-rewriting systems. Thus, we are able to formulate the conditions of [8] in the context of string rewriting systems.

A string rewriting system  $R$  is *quasi-deterministic* if and only if

1. No rule has  $\lambda$  as its right-hand side
2. No rule in  $R$  is *end-stable*—i.e., no rule has the same rightmost symbol on its left- and right-hand sides, and
3.  $R$  has no *end pair repetitions*—i.e., no two rules in  $R$  have the same unordered pair of rightmost symbols on their sides.

We define a *right-hand-side critical pair* as follows: if  $l_1 \rightarrow r_1$  and  $l_2 \rightarrow r_2$  are two distinct rewrite rules and  $r_2 = xr_1$  for some  $x$  (i.e.,  $r_1$  is a suffix of  $r_2$ ) then  $\{xl_1, l_2\}$  is a right-hand-side critical pair. The set of all right-hand-side critical pairs is referred to as  $\text{RHS}(R)$ .

It can be shown [6] that

**Lemma 3.1.** *Suppose  $R$  is a convergent quasi-deterministic string rewriting system. Then  $RHS(R)$  is not quasi-deterministic if and only if  $RHS(R)$  has an end pair repetition.*

A string-rewriting system is *deterministic* if and only if it is non-subterm-collapsing and  $RHS(R)$  is quasi-deterministic.

A *Lynch-Morawska string rewriting system* or *LM-system* is a convergent right-reduced string rewriting system  $R$  which satisfies the following conditions:

- (i)  $R$  is non-subterm-collapsing,
- (ii)  $R$  is forward-closed, and
- (iii)  $RHS(R)$  is quasi-deterministic.

In light of the results of [7], a convergent string rewriting system  $R$  is an LM-system if and only if  $RHS(R)$  is quasi-deterministic and

- (a)  $R$  is almost-left reduced (see [7]).
- (b) There are no overlaps among the left-hand sides of  $R$ .
- (c) No lhs overlaps with a rhs.

We now work towards proving the main results of this paper. Namely, we will show in the sequel below that the subterm-collapse problem for convergent and forward-closed string rewriting systems is decidable.

The first of our results towards the above goal is below:

**Lemma 3.2.** *Let  $R$  be a convergent, forward-closed and quasi-deterministic string rewriting system and  $x, y, z \in IRR(R)$  such that  $xy \rightarrow^! z$ . Then there exist irreducible strings  $x = x_1, x_2, \dots, x_n, x_{n+1}$ ,  $y_1, y_2, \dots, y_n, y_{n+1}$  such that*

- 1.  $y = y_1 \dots y_{n+1}$ ,
- 2.  $x_i y_i$  is an innermost redex for all  $1 \leq i \leq n$ ,
- 3.  $x_i y_i \rightarrow x_{i+1}$  for all  $1 \leq i \leq n$ , and
- 4.  $x_{n+1} y_{n+1} = z$ .

*Proof.* The proof proceeds by induction on the number of rewrite steps along the path from  $xy$  to the normal form  $z$ .

**Basis.** Suppose  $x, y, z \in IRR(R)$  such that  $xy \rightarrow^! z$  in  $k = 1$  steps. That is,  $xy \rightarrow z$ . Since  $x, y \in IRR(R)$  there cannot be a redex that is a substring of either  $x$  or  $y$  alone. Hence there must be strings  $x', y' \in \Sigma^*$  and  $l_1, l_2 \in \Sigma^+$  such that

$$x = x' l_1, y = l_2 y'$$

and  $l_1 l_2 = l$  for some  $l \rightarrow r \in R$ . Note that, since  $R$  is convergent we may assume that  $x' l_1 l_2$  is the shortest such redex.

$\therefore$  We can construct the following sequence:  $x_1 = x' l_1$ ,  $x_2 = x' r$ ,  $y_1 = l_2$ ,  $y_2 = y'$  such that,

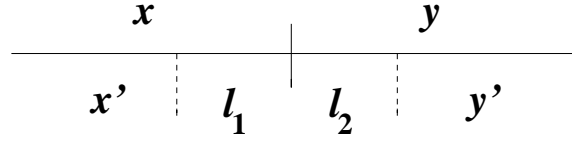


Figure 2: first step

- (1)  $y = y_1 y_2 = l_2 y'$
- (2)  $x_1 y_1 = x' l_1 l_2 = x' l$  is an innermost redex
- (3)  $x_1 y_1 \rightarrow x_2 = x' r$
- (4)  $x_2 y_2 = x' r y' = z$

Above, (1) and (3) follow immediately from the definitions of  $x_1, y_1, x_2, y_2$ . (4) will follow after establishing (2). However, since  $x' l_1 l_2$  was chosen as the shortest such redex appearing in  $xy$  from the left and crossing the boundary between  $x$  and  $y$ , it must be an innermost redex. Thus, we have established (2).

Now, since  $R$  is forward-closed,  $x' r$  can be assumed to be in normal form. Since  $y'$  is a proper suffix of  $y \in \text{IRR}(R)$ , we get that  $y' \in \text{IRR}(R)$ . Note that the above reductions are leftmost-largest. Since every string has a unique leftmost-largest normal form modulo a terminating string-rewriting system, and since  $R$  is convergent, this normal form must be  $z$ .

**Inductive Hypothesis.** Assume that the result holds for all  $x, y, z \in \text{IRR}(R)$  such that  $xy \rightarrow^! z$  in  $k > 1$  steps. We show that it holds for strings  $x, y, z \in \text{IRR}(R)$  such that  $xy \rightarrow^! z$  in  $k + 1$  steps.

Since  $k > 1$ ,  $\exists w \in \Sigma^+$  such that  $xy \rightarrow w \rightarrow^! z$ . Note that  $w \rightarrow^! z$  must take exactly  $k$  rewrite steps. As in the base case, since  $x, y \in \text{IRR}(R)$  and  $xy$  is reducible, we have that  $xy = x' l_1 l_2 y'$  where  $x = x' l_1$ ,  $y = l_2 y'$ , and  $l_1 l_2 = l$  for some  $l \rightarrow r \in R$  and  $x', y' \in \Sigma^*$ . Since  $R$  is convergent, we assume that  $x' l$  is the leftmost prefix of  $xy$  that is a redex.

We thus form the sequence:  $x_1 = x' l_1$ ,  $y_1 = l_2$ ,  $x_2 = x' r$ ,  $y_2 = y'$ . Since  $x_1 y_1$  is the leftmost redex of  $xy$  it must be the case that  $x_1 y_1$  is an innermost redex. Therefore,  $x' r$  can be assumed to be in normal form. Then  $w = x' r y'$ , and since  $x' r \in \text{IRR}(R)$  and  $y \in \text{IRR}(R)$  we get that  $w = uv$  for some  $u, v \in \text{IRR}(R)$ . We can then apply the induction hypothesis to  $u, v$ , and  $z$  to fill in the rest of the sequence with the desired properties.

Therefore we can conclude that the result holds for all  $x, y, z \in \text{IRR}(R)$  such that  $xy \rightarrow^! z$ .  $\square$

An immediate consequence of the definition of subterm-collapse given below.

**Lemma 3.3.** *Let  $R$  be a convergent forward-closed string rewriting system and  $x, y \in \text{IRR}(R)$  such that  $xy \rightarrow^! x$  and  $y \neq \lambda$ . (Thus  $x$  causes a subterm-collapse.) Let  $y_1$  be a prefix of  $y$ . Then  $xy_1$  causes a subterm-collapse.*

*Proof.* Let  $x, y \in \text{IRR}(R)$  such that  $xy \rightarrow^! x$ . Let  $\tilde{y}$  be any prefix of  $y$ . Thus  $y = \tilde{y} y'$  for some string  $y'$ .

In order to generate a subterm-collapse with respect to  $x\tilde{y}$ , we must have a string  $w \in \Sigma^+$  such that  $x\tilde{y}w \rightarrow^* x\tilde{y}$ . We construct such a string as follows: let  $w = y' \tilde{y}$ .

Therefore,  $x\tilde{y}w = x\tilde{y} y' \tilde{y} = xy\tilde{y}$ . Since  $xy \rightarrow^! x$  we get  $x\tilde{y}w = xy\tilde{y} \rightarrow^* x\tilde{y}$ .  $\square$

We now prove that  $R$  is subterm-collapsing if and only if there is a right-hand side of a rule in  $R$  that causes a subterm collapse in the sense of the above definition. This lemma will be key in showing the decidability of the subterm-collapse problem as it allows us only to consider right-hand sides of rules for possible sources of subterm-collapse.

**Lemma 3.4.** *Suppose  $R$  is a convergent forward-closed string rewriting system. Then  $R$  is subterm-collapsing if and only if there is a right-hand side  $r$  that causes a subterm-collapse.*

*Proof.* If there is a right-hand side that causes a subterm-collapse, then  $R$  is subterm-collapsing. Towards proving the “only if” direction, assume for the sake of deriving a contradiction that the result doesn’t hold, i.e.,  $R$  is subterm-collapsing but no right-hand side causes a subterm-collapse. Then, let  $w$  be one of the *shortest* strings that causes a subterm-collapse.

Since  $w \neq \lambda$  it must be the case that  $(\exists a \in \Sigma)(\exists w' \in \Sigma^*) : w = aw'$ . Also, since  $w$  is assumed to cause a subterm-collapse,  $(\exists z \in \Sigma^+) : wz = aw'z \rightarrow^* w = aw'$ . There are thus two cases to consider: either  $a$  is involved in the reduction, i.e.,  $a$  is in the  $l$ -part of a redex, or it is not.

Suppose  $a$  is involved in the reduction. By Lemma 2.2, without loss of generality we can assume that  $R$  is right-reduced. Since  $a$  is the first letter of  $w$  and  $a$  is involved in some reduction step, there must be a prefix  $z'$  and a corresponding suffix  $z''$  of  $z$  such that  $wz' = aw'z' \rightarrow^* aw'' \rightarrow r$  and  $rz'' \rightarrow^* w$  for some  $w''$ . That is,  $aw''$  is a redex as well as its  $l$ -part, i.e.,  $aw'' = l$  for some  $l \rightarrow r \in R$ . But by the previous lemma (Lemma 3.3),  $wz'$  and hence  $r$  causes a subterm-collapse. This contradicts our assumption.

Now, suppose  $a$  is not involved in the reduction sequence. Then it must be that  $w'z \rightarrow^* w'$ . Thus,  $w'$  causes a subterm-collapse and  $|w'| < |w|$ , which contradicts the minimality of  $w$ .  $\square$

The main lemma of this section appears below. It gives us that a certain language, parameterized by two strings  $u, v \in \Sigma^*$ , is a deterministic context-free language. We prove this by constructing a deterministic pushdown automaton to recognize this language.

**Lemma 3.5.** *Let  $R$  be a convergent, right-reduced, and forward-closed string rewriting system,  $u, v \in IRR(R)$ , and  $\# \notin \Sigma$ . Then the language*

$$\mathcal{L}_{u,v} = \left\{ w\# \mid uw \rightarrow^! v, w \neq \lambda \right\}$$

*is a deterministic context-free language over  $(\Sigma \cup \{\#\})^*$*

*Proof.* We design a deterministic pushdown automaton (DPDA)  $\mathcal{M}$  that recognizes  $\mathcal{L}_{u,v}$ . In the sequel, let  $x$  denote the contents of  $\mathcal{M}$ ’s stack from bottom to top.

Initially,  $\mathcal{M}$  pushes a special symbol,  $\$$ , onto the stack (which serves as a bottom marker) and then pushes  $u$ . Thus, the contents of the stack after the initialization steps are  $\$u$ .

Then, we design a transition system based on two cases. Either pushing the symbol  $a \in \Sigma$  completes a redex or it does not. That is,

- 1  $(x, a) \mapsto (xa, \lambda)$  if  $xa$  is not a redex, or
- 2  $(x, a) \mapsto (x'r_0, \lambda)$  if  $xa = x'l_0$  where  $x'$  is the  $s$ -part and  $l_0$  the  $l$ -part of  $xa$  (i.e.,  $l_0$  the *longest* left-hand side in  $R$  that is a suffix of  $xa$ ).



$\mathcal{M}$  will carry out the above transitions by pushing symbols of  $w$  (which is initially on the tape) and reducing each redex that appears. When  $\mathcal{M}$  reaches the  $\#$  symbol, by Lemma 3.2 if  $uw \rightarrow^! v$  then the contents of the stack must be  $\$v$ . This can be checked by  $\mathcal{M}$ .

Finally,  $\mathcal{M}$  can be created by building an Aho-Corasick automaton,  $\mathcal{K}$ , for the set  $\{l_1, l_2, \dots, l_n\}$  as given, for instance, in [5]. Then  $\mathcal{M}$  can simulate  $\mathcal{K}$  on its stack by essentially restarting  $\mathcal{K}$  whenever  $\mathcal{K}$  accepts.  $\square$

As a consequence of the above Lemma 3.5 the subterm collapse problem is decidable for convergent, forward-closed, string-rewriting systems.

**Corollary 3.6.** *The following decision problem:*

Given: *A convergent, forward-closed, right-reduced SRS  $R$ .*

Question: *Is  $R$  subterm-collapsing?*

*is effectively solvable.*

*Proof.* A decision procedure can be constructed by creating, for each  $l \rightarrow r \in R$ , a DPDA  $\mathcal{M}_r$  such that  $L(\mathcal{M}_r) = \mathcal{L}_{r,r}$  by lemma 3.5.  $\mathcal{M}_r$  can then be converted into an equivalent context-free grammar  $G_r$ . Then  $L(G_r) = \emptyset$  if and only if  $r$  does not cause a subterm-collapse. By Lemma 3.4 this is enough to conclude that  $R$  is not subterm-collapsing in general. Finally, deciding if a CFG generates the empty language is decidable, therefore, the overall problem is decidable as well.  $\square$

Note also, that the construction outlined above can be carried out *in polynomial time*. Thus, not only is the above subterm-collapse problem for convergent, forward-closed string rewriting systems decidable, it is efficiently decidable. This is in contrast to the results of [6] where it was shown that checking if a given term-rewriting system is subterm-collapsing, even when the system satisfies all of the other Lynch-Morawska conditions, is undecidable.

We can therefore conclude that the problem of verifying if a convergent and forward-closed string rewriting system (or a term rewriting system over a signature of monadic function symbols) is an *LM*-system is decidable.

As another corollary of the above result, we get that the cap problem for convergent, forward-closed, string-rewriting systems is also decidable. This problem, also known as the deduction problem, is often studied in the field of symbolic cryptographic protocol analysis.

**Corollary 3.7.** *The Cap Problem:*

Given: *A convergent, forward-closed string-rewriting system  $R$ , a string  $u \in \Sigma^+$  (representing the intruder knowledge) and a secret  $v \in \Sigma^+$ .*

Question: *Does there exists a string  $w \in \Sigma^+$  (called a cap term) such that  $uw \rightarrow_R^! v$ ?*

*is decidable.*

*Proof.* The construction is essentially the same as that in the proof of Corollary 3.6. This time a DPDA is constructed, using Lemma 3.5, for the language  $\mathcal{L}_{u,v}$ .  $\square$

The result of Corollary 3.7 is contrasted with the fact that, for general term-rewriting systems, the cap problem is known to be undecidable. The cap problem remains undecidable even when restricted to  $LM$ -Systems. The above results shows, in the monadic case, if  $R$  is convergent and forward-closed, then the problem becomes decidable.

## References

- [1] Siva Anantharaman, Paliath Narendran, and Michael Rusinowitch. Intruders with caps. In *Proceedings of the 18th international conference on Term rewriting and applications*, pages 20–35. Springer-Verlag, 2007.
- [2] Siva Anantharaman, Paliath Narendran, and Michael Rusinowitch. String rewriting and security analysis: an extension of a result of Book and Otto. *Journal of Automata, Languages and Combinatorics*, 16(2–4):83–98, 2012.
- [3] Franz Baader and Tobias Nipkow. *Term rewriting and all that*. Cambridge university press, 1999.
- [4] Ronald V Book and Friedrich Otto. *String-rewriting systems*. Springer, 1993.
- [5] Maxime Crochemore and Wojciech Rytter. *Text algorithms*, volume 698. Oxford University Press.
- [6] Kimberly Gero, Chris Bouchard, and Paliath Narendran. Some notes on basic syntactic mutation. In Santiago Escobar, Konstantin Korovin, and Vladimir Rybakov, editors, *UNIF 2012 Post-Worskhop Proceedings. The 26th International Workshop on Unification*, volume 24 of *EPiC Series in Computing*, pages 17–27. EasyChair, 2014.
- [7] Daniel S. Hono II, Namrata Galatage, Kimberly A. Gero, Paliath Narendran, and Ananya Subburathinam. Notes on lynch-morawska systems. Technical Report SUNYA-CS-16-01, Department of Computer Science, University at Albany—SUNY, 2016.
- [8] Christopher Lynch and Barbara Morawska. Basic syntactic mutation. In *Automated Deduction (CADE-18)*, pages 471–485. Springer, 2002.
- [9] Gérard Sénizergues. A polynomial algorithm testing partial confluence of basic semi-thue systems. *Theoretical computer science*, 192(1):55–75, 1998.